Group Project Report

Advanced Business Analytics

Prof. Murtaza Nasir

Barton School of Business

WICHITA STATE UNIVERSITY

Project 2

Group 10
Vaishnavi Mandadi(D628T797)
Pratiksha Kunwar(W743J246)
Sumanth Reddy Peddavandla (S293V266)

Table of Contents

Gasoline Blending Problem

Problem Description

Jansen Gas creates three types of aviation gasoline(avgas), labeled A, B, and C. It does this by blending four feedstocks: Alkylate; Catalytic Cracked Gasoline; Straight Run Gasoline; and Isopentane. Jansen's production manager, Dave Wagner, has compiled the data on feedstocks and gas types in Tables 4.6 and 4.7. Table 4.6 lists the availability and values of the feedstocks, as well as their key chemical properties, Reid vapor pressure, and octane rating. Table 4.7 lists the gallons required, the prices, and chemical requirements of the three gas types.

Table 4.6 Data on Feedstocks

Feedstock	Alkylate	CCG	SRG	Isopentane
Gallons available (1000s)	140	130	140	110
Value per gallon	$4.50	$2.50	$2.25	$2.35
Reid vapor pressure	5	8	4	20
Octane (low TEL)	98	87	83	101
Octane (high TEL)	107	93	89	108

Table 4.7 Data on Gasoline

Gasoline	A	B	C
Gallons required (1000s)	120	130	120
Price per gallon	$3.00	$3.50	$4.00
Max Reid pressure	7	7	7
Min octane	90	97	100
TEL level	Low	High	High

Note that each feedstock can have either a low or a high level of TEL, which stands for tetraethyl lead. This is measured in units of milliliters per gallon, so that a low level might be 0.5 and a high level might be 4.0. (For this problem, the actual numbers do not matter.) As indicated in Table 4.6, the TEL level affects only the octane rating, not the Reid vapor pressure. Also, gas A is always made with a low TEL level, whereas gas types B and C are always made with a high TEL level. As indicated in Table 4.7, each gasoline has two requirements: a maximum allowable Reid vapor pressure and a minimum required octane rating. In addition to these requirements, the company wants to ensure that the amount of gas A produced is at least as large as the amount of gas B produced. Dave believes that Jansen can sell all the gasoline it produces at the given prices. If any feedstocks are left over, they can be sold for the values indicated in Table 4.6. He wants to find a blending plan that meets all the requirements and maximizes the revenue from selling gasoline and leftover feedstocks. To help Dave with this problem, you should develop an LP optimization model and then use pulp in python to find the optimal blending plan and maximum Revenue.

Approach:
In this problem, we have four feedstocks available: Alkylate, CCG, SRG, and Isopentane, each with a different amount available, a value per gallon, and specific Reid vapor pressure and octane rating values. We also have three gasoline types, A, B, and C, with a specific number of gallons required, a price per gallon, a maximum Reid vapor pressure, and a minimum octane rating.

Objective:
The objective is to maximize the total revenue obtained from producing and selling gasoline blends, subject to constraints on the properties of the blends and the availability of the feedstocks.

Mathematical Formulation
To solve the gasoline blending problem, we used linear programming to formulate an optimization model with decision variables representing the amounts of each feedstock used to produce each gasoline blend. We can also calculate the amount of each feedstock leftover after blending and the amount of each gasoline blend produced for sale. The objective function maximizes the total revenue from selling gasoline blends, which includes revenue from selling the gasoline blends themselves and revenue from selling leftover feedstocks. The constraints in the problem include limits on the Reid vapor pressure and octane rating of each gasoline blend, as well as minimum production requirements for each gasoline blend and limits on the total amount of feedstock that can be used.

Code Explanation
Step 1: Import required libraries and set up data.

```
✓ 11s  !pip install pulp

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pulp
  Downloading PuLP-2.7.0-py3-none-any.whl (14.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.3/14.3 MB 48.0 MB/s eta 0:00:00
Installing collected packages: pulp
Successfully installed pulp-2.7.0
```

```
✓ 0s  [2]  from pulp import *
           import pandas as pd
```

First, we imported the pulp library, which provides tools for formulating and solving linear programming problems. We also imported the pandas library, which is used to work with data.

Data

Add data

```
[3]  # data
     feedstocks = ['Alkylate', 'CCG', 'SRG', 'Isopentane']
     gas_types = ['A', 'B', 'C']

     #feedstocks data
     gallons_available = dict(zip(feedstocks, [140000, 130000, 140000, 110000]))
     value_per_gallon = dict(zip(feedstocks, [4.50, 2.50, 2.25, 2.35]))
     reid_vapor_pressure = dict(zip(feedstocks, [5, 8, 4, 20]))
     octane_rating_low = dict(zip(feedstocks, [98, 87, 83, 101]))
     octane_rating_high = dict(zip(feedstocks, [107, 93, 89, 108]))

     #Gas data
     gallons_required = dict(zip(gas_types, [120000, 130000, 120000]))
     price_per_gallon = dict(zip(gas_types, [3.00, 3.50, 4.00]))
     max_reid_pressure = dict(zip(gas_types, [7, 7, 7]))
     min_octane_rating = dict(zip(gas_types, [90, 97, 100]))
```

Next, we defined two lists' feedstocks and gas types that represent the types of feedstocks and gasoline being used. We then defined dictionaries that provide data about each feedstock and gasoline type, including the number of gallons available, the value per gallon, the reid vapor pressure, and the octane rating.

Step 2: Define the LP problem.

Programming

Formulate Linear Programming (LP) Problem

```
[4]  # problem definition
     prob = LpProblem('GasolineBlendingProblem', LpMaximize)
```

Next, we defined a LpProblem object named prob that represents the linear programming problem we want to solve. We set the name of the problem to "Gasoline Blending Problem" and set the optimization objective to maximize revenue.

Step 3: Define decision variables.

```
[23] #Decision Variables
     variable = LpVariable.dicts("gasoline", (feedstocks, gas_types), 0, None, LpContinuous)
```

We defined a decision variable named variable using the LpVariable.dicts() method. This creates a dictionary of LP variables that represent the number of gallons of each feedstock needed to produce each type of gasoline. We set the lower bound of the variables to zero and do not set an upper bound (i.e., they are unbounded).

Step 4: Calculate leftover feedstock and sell gasoline.

```
#Leftover Calculation
Leftover = {f: gallons_available[f] - lpSum(variable[f]) for f in feedstocks}
```

```
[26]
#Selling gasoline Calculation
sell_gas= {g: lpSum(variable[f][g] for f in feedstocks) for g in gas_types}
```

We calculated the leftover amount of each feedstock by subtracting the sum of the feedstock used for each gasoline type from the total amount available. We also calculated the amount of each gasoline type that will be sold by summing the amount of each feedstock used to produce that gasoline type.

Step 5: Define the objective function.

Objective Function

```
[27] #Objective function for Revenue
prob += lpSum(price_per_gallon[g] * sell_gas[g] for g in gas_types) + lpSum(value_per_gallon[f] * Leftover[f] for f in feedstocks), "Total Revenue"
```

We defined the objective function for the LP problem by summing the revenue generated by selling each gasoline type and the value of any leftover feedstocks. We used the lpSum() function to calculate the sum of the products of the price per gallon and the number of gallons sold for each gasoline type and the value per gallon and the number of gallons leftover for each feedstock.

Step 6: Define the constraints.

We defined several constraints for the LP problem.

+ Code + Text

```
[28] #ReidVapour Constraints
for g in gas_types:
    prob += (variable[f][g]* reid_vapor_pressure[f] for f in feedstocks) <= lpSum(variable[f][g]*max_reid_pressure[g] for f in feedstocks)
```

The first set of constraints ensures that the reid vapor pressure of each gasoline type does not exceed the maximum allowed value, which is determined by the reid vapor pressure of the feedstocks used to produce the gasoline type.

```
#Octane Rating Constraints
for g in gas_types:
    if g == 'A':
        prob += lpSum([variable[f][g] * octane_rating_low[f] for f in feedstocks]) >= lpSum(variable[f][g] * min_octane_rating[g] for f in feedstocks)
    else:
        prob += lpSum([variable[f][g] * octane_rating_high[f] for f in feedstocks]) >= lpSum(variable[f][g] * min_octane_rating[g] for f in feedstocks)
```

The second set of constraints ensures that the octane rating of each gasoline type meets the minimum required value, which is different for each gasoline type.

```
#Gas A produced is atleast as large as Gas B
prob += lpSum(variable[f]["A"] for f in feedstocks) >= lpSum(variable[f]["B"] for f in feedstocks)
```

The third constraint ensures that the amount of Gas A produced is at least as large as the amount of Gas B produced.

```
#Total Gallons made greater then Required
for g in gas_types:
    prob += lpSum(variable[f][g] for f in feedstocks) >= gallons_required[g]
```

The fourth constraint ensures that the total number of gallons produced of each gasoline type is greater than or equal to the number of gallons required for that gasoline type.

```
[32] #Total Used should be less than Availability
     for f in feedstocks:
         prob += lpSum(variable[f]) <= gallons_available[f]
```

The fifth constraint ensures that the total amount of each feedstock used does not exceed the amount available.

Step 7: Solve the LP problem.
We used the solve method of the prob object to solve the LP problem.

Run Optimization Algorithm

```
[33] prob.solve()

     1
```

Step 8: Print the results

Get Status & Results

```
[34] print("Status:", LpStatus[prob.status])

     Status: Optimal
```

```
for v in prob.variables():
    print(v.name, "=", v.varValue)
```

```
gasoline_Alkylate_A = 37131.064
gasoline_Alkylate_B = 34312.268
gasoline_Alkylate_C = 68556.668
gasoline_CCG_A = 87887.023
gasoline_CCG_B = 0.0
gasoline_CCG_C = 0.0
gasoline_Isopentane_A = 82.552639
gasoline_Isopentane_B = 22230.483
gasoline_Isopentane_C = 24772.578
gasoline_SRG_A = 4899.36
gasoline_SRG_B = 73457.249
gasoline_SRG_C = 61643.391
```

```
[36] print("Total revenue = ", value(prob.objective))

     Total revenue =  1718021.7973653502
```

We use the LpStatus function to print the status of the LP problem (i.e., whether it was solved successfully). We then use a loop to print the values of each LP variable, which represent the number of gallons of each feedstock needed to produce each gasoline type. Finally, we print the total revenue generated by the optimal solution of the LP problem.

Sensitivity Analysis:

```
a = [{'name':name,'shadow price':g.pi,'slack': g.slack} for name, g in prob.constraints.items()]

print(pd.DataFrame(a))
```

	name	shadow price	slack
0	_C1	-0.272305	-0.000000e+00
1	_C2	-0.240019	-0.000000e+00
2	_C3	-0.240019	-0.000000e+00
3	_C4	-0.222690	-0.000000e+00
4	_C5	-0.183782	7.909289e-12
5	_C6	-0.183782	7.163692e-12
6	_C7	-0.440377	-0.000000e+00
7	_C8	-0.000000	-1.000000e+04
8	_C9	-0.389032	-0.000000e+00
9	_C10	-0.000000	-3.497264e+04
10	_C11	1.266511	-0.000000e+00
11	_C12	-0.000000	4.211298e+04
12	_C13	0.448460	-0.000000e+00
13	_C14	-0.000000	6.291439e+04

This output shows the sensitivity analysis of the optimization problem, where each constraint has been analyzed for its shadow price and slack value.

The table shows the shadow prices and slack values of the constraints in the optimization model. Shadow prices represent the increase in objective function value per unit increase in the right-hand side of the constraint, while slack values represent the unused amount of the resource or capacity of each constraint. The negative shadow prices suggest that increasing the corresponding constraint would not increase the objective function value, while the positive shadow prices indicate that relaxing or removing the constraint would result in a decrease in the objective function value. The slack values of most constraints are very small, which means that the resources are almost fully utilized. However, constraints _C8 and _C10 have relatively large slack values, which suggests that there is some unused capacity in the corresponding resources.

Conclusion:
The maximum revenue that Jansen Gas can generate by blending the feedstocks according to the optimal plan is **$1,718,021.80.**